Senior Design Group #0911

Google Calendar

By:
Jeff Forshée, Marco Murillo, and Lee Hinsz

Advisor: Dr. You

Client: ECE Department

Table of Contents

Technician's Troubleshooting Section:

Project Comments:

Appendix:

Introduction:

The project will be a stand-alone display for Google calendar that will display various data from the Google calendar Application Program Interface (API). It is originally intended to help with professor's schedules within the ECE department. This would be useful for setting up appointments and providing information to students. Our original client is the ECE department, but the project will be able to be used in any environment within proximity of internet access and may have wider commercial applications.

Objectives:

- The unit will operate on any standard wireless network
- The unit will receive and display data from the Google calendar API
- The unit will be able to display the events of the day, week and month
- The user will have the ability to change user settings without reprogramming the device
- The device will be able to use any VGA or HDMI monitor
- The unit should be 'stand alone' and require power as its only physical connection
-

Previous Work:

Others work:

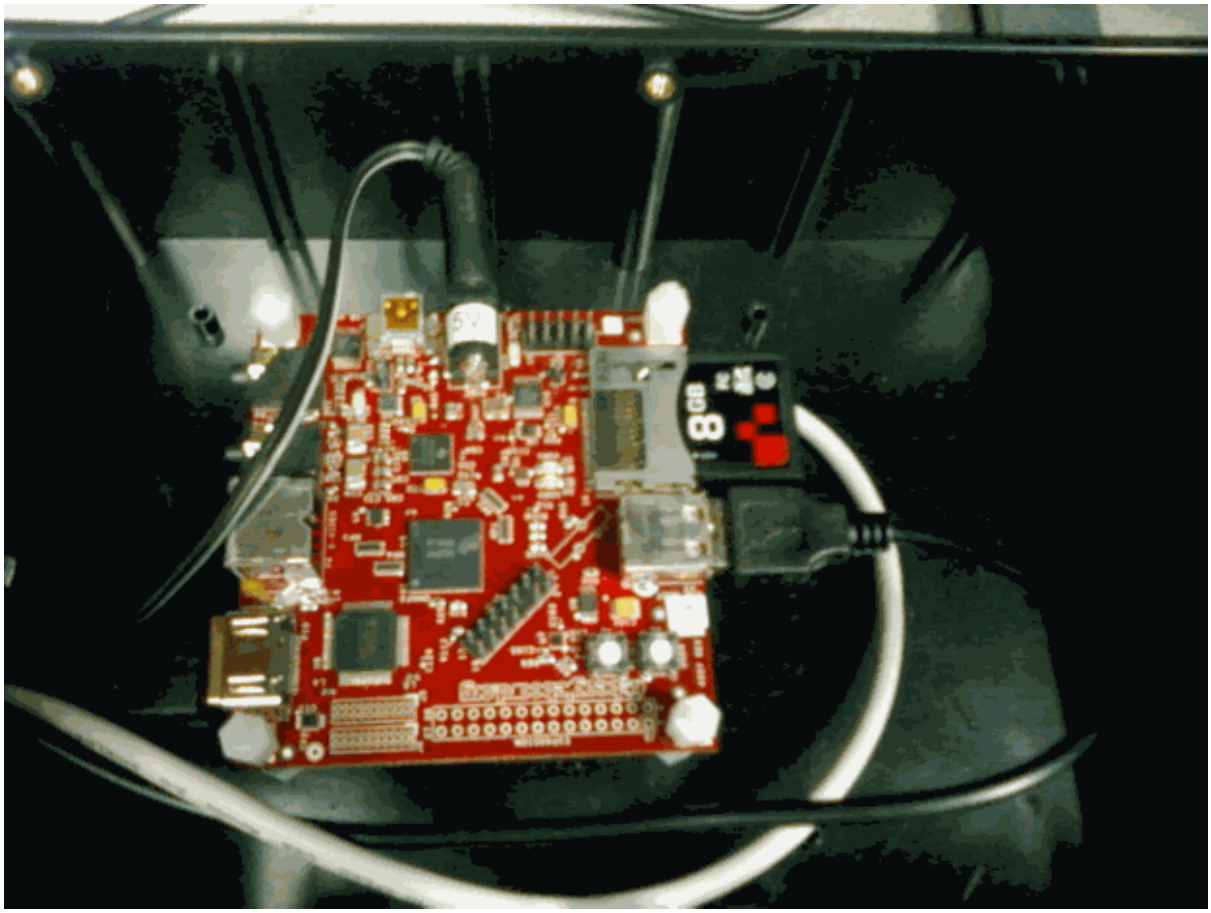
Google has built a prototype called the Radish, which is very similar to what we are trying to accomplish with our project. Their project has the goals of efficiency and eliminating paper. Their application was to update schedules for their conference rooms. They use a radio frequency to ping a server that will update information from Google Calendar for their device. The Radish is ultra power efficient with a LCD to display the schedule, a XBEE protocol to receive data from a server, and an indoor solar panel to power the device.

There is also DIY project called "Larmie" that has been posted online that uses an Arduino microprocessor package as an interface between a PC and an alarm clock. The PC hosts the information from Google Calendar to the Arduino which then interprets the data, constantly updating the alarm clock and setting off alarms for each scheduled event on the calendar.

Our work:

Prior to this semester we worked with 2 different development board in attempts to try to get a patent for our work, and to maybe start a company that would manufacture the devices. The first semester we developed a working concept, but was not able to get a graphical calendar due to the limitations of the hardware. The second semester we upgraded to some better technology and hardware. However the lack of documentation for programming, and memory limitations for getting our scripting language to operate where still a problem. Lastly we came to the conclusion to ditch the FPGA idea, and work on something that is more linux friendly. Thus we continued the project with the Beagle Board.

Labeled pictures of device:



Design Options and Selected Approach:

We talked to Dr. You and one of his design requirements was that we use an FPGA for our project. Dr. You also wanted the device to use an LCD monitor. Also, due to our budget constants, Dr. You also recommended that we use equipment that is available within the department. The department had two FPGA development boards available, the Altera Cyclone II and the XILINK Spartan 3-E.

We looked at specs of each board and the support software for each board, and found that the Altera FPGA would offer the best options for completing this project. The Altera FPGA board has a C to HDL compiler. This will be highly useful and allow us to avoid most of the VHDL and Verilog programming. This should also greatly reduce the development time making the project more feasible. The Cyclone II is also a much more recent chip, has a lot more logic elements, and has more recent support documentation and community support. The Spartan E-3 has almost none of this since is has been discontinued so finding support would be far more difficult.

Device	EP2C35 (Altera)	XC3S50A (XILINX)
Logic Elements	33126	1584
Total RAM bits	119808	54000
Dedicated Multipliers (18x18)	13	3

It is apparent from the table that Altera FPGA is a much more advanced chip then the Xilinx offering. The massive increase in the number of logic elements will allow for a much more sophisticated program and may allow us to add additional features if time permits.
Both of the boards have fairly comparable support hardware. Both boards have a VGA port and an Ethernet port. While both boards fulfill our interface requirements the Altera does have some additional hardware that the Xilinx does not. With either of the boards have the ability to use any VGA monitor or touch screen display and interface with the internet.

Since we have the option with our Altera compiler to go from C to HDL, we will chose to program in the C programming language. The higher level of this language will allow us to use more sophisticated logic more easily. There is a chance we will need to do some programming in a HDL in which case we would prefer Verilog over VHDL due to its relative simplicity and superior support on our hardware.

On the recommendation of Dr. You, we will start by working on a simplified prototype. The simplified prototype will interface with the internet using the on-board support hardware and will not utilize a wireless connection. The wireless functionality will be added in the second semester along with the user interface.

After developing our project on an FPGA and facing many constraints, like memory and speed. We decided to switch completely our hardware, and start developing on a BeagleBoard.

BeagleBoard Features:

No.	Name	Comment
1	OMAP3530 processor + 256MB NAND + 128MB DDR (rev B) + 256MB DDR (rev C)	PoP: Package-On-Package implementation for Memory Stacking

		256MB NAND/128MB Mobile DDR SDRAM available from DigiKey (512MB NAND/256MB Mobile DDR SDRAM available from DigiKey) Micron's multi chip packages (MCPs) for Beagle Board
2	DVI chip (TFP410)	
3	DVI-D	Connection via HDMI connector
4	14-pin JTAG	1.8V only!
5	Expansion connector: I2C, I2S, SPI, MMC/SD	User must solder desired header into place
6	User button	Allows setting boot order.
7	Reset button	
8	USB 2.0 EHCI HS	Rev A and B: not working, unpopulated Rev C: populated and working
9	SD/MMC+	SDHC cards are supported
10	RS-232 serial	
11	Alternate power	normally powered by USB (unmounted on REV Ax boards, see errata)
12	USB 2.0 HS OTG	Mini-AB connector. Board can be powered from port.
13	Stereo In	
14	Stereo Out	
15	S-Video	
16	TWL4030 (Rev A thru C2 inc.) TPS65950 (Rev C3 onwards)	Audio CODEC, USB port, power-on reset and power management. The TWL4030 is pin-compatible with the TPS65950 chip and was used due to the very limited availability of the TPS65950 in early board revisions.
17	LCD	only rev C
18	USB power	
19	Host PHY	
20	32kHz	
21	12MHz	
22	RS232 XVCR	
23	PWR SW	
24	VBAT	

- Board size: 3" x 3" (about 76.2 x 76.2 mm)
- Weight: ~37g

- Currently 6 layer PCB; target: 4 layer

Budget Over all Semesters:

Part	**Price**
Displays (LCD, LCD touch screen)	$0 (borrowed from the department)
Development Board Altera DE-2	$0 (board owned by the department)
Software (C to HDL compiler)	$0 (software comes with development board)
Ethernet cord	$15
Nios II Embedded Evaluation Kit (NEEK), Cyclone III Edition	$450
Beagle board	$150
Mouse	Supplied by Department
Monitor	Supplied by Department
Ethernet adapter	$15
Enclosure	Supplied by Department
Keyboard	Supplied by Department
Wireless Dongle	$15
CAT 5 cable	Supplied by Department
Flash Card	$20
5 VDC Power Supply	$8
Beagle board	$150
Mouse	Supplied by Department
Ethernet adapter	$8
Literature	$30 (software manuals, printing allocations)

Requirements Capture

Introduction:

Our goal is to develop a stand-alone display for Google calendar that will display various data from the Google calendar Application Program Interface (API) or failing that, our own custom API. It is originally intended to help with professor's schedules within the ECE department, but it can be used for wider applications. We intend to implement this design using a Field Programmable Gate Array (FPGA) and any necessary support hardware.

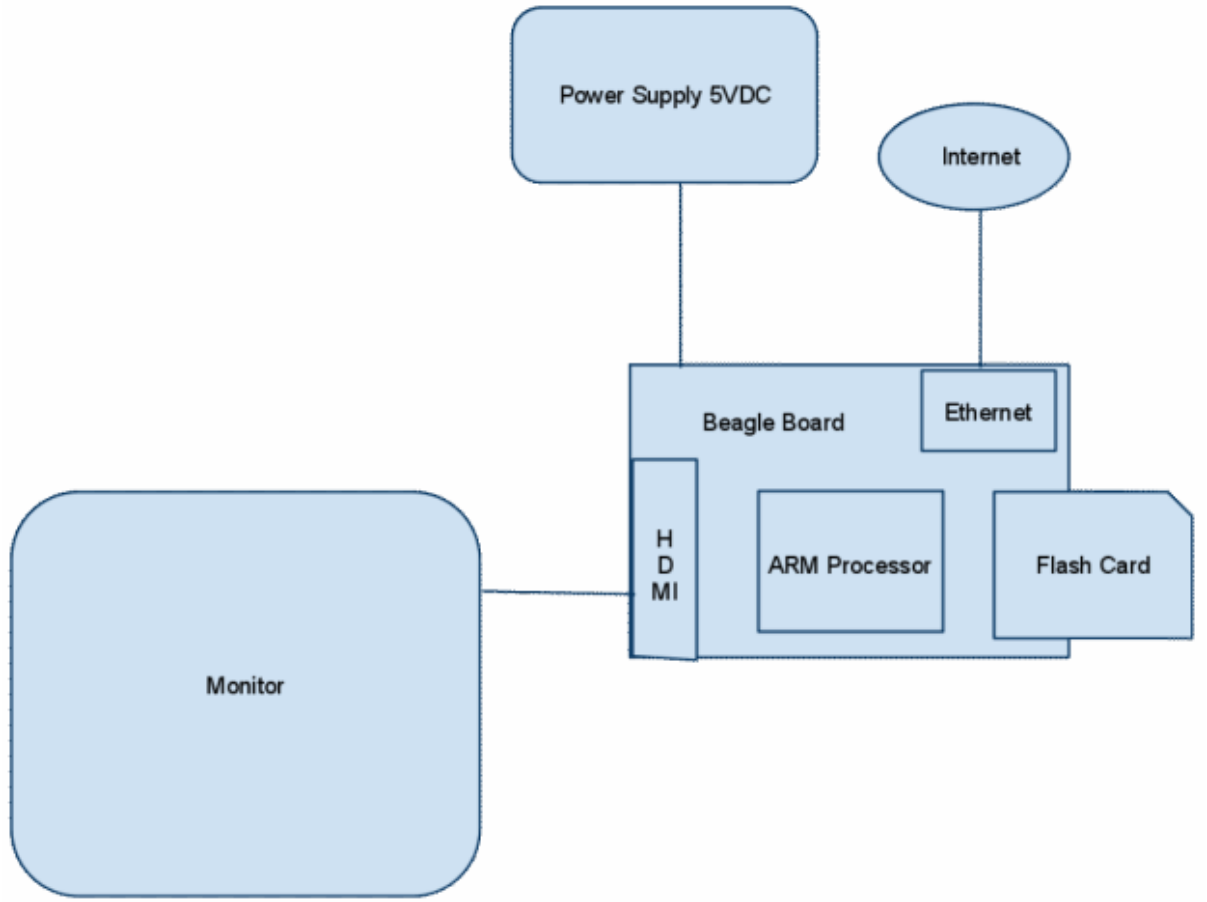
Requirements:

• The unit must be wireless and interface with the school's network.
• The unit must retrieve data from Google calendar as well as display calendar data.
• The display will be able to scroll through the events of the day, week, and month.
• The design will be executed with an FPGA.
• The user will have the ability to change calendars (without reprogramming the FPGA).
• The device will use the board's integrated touch screen.
• The unit should be 'stand-alone' and should rely on no other devices to operate.
• The unit will have a touch screen interface that will allow the user to change settings.
• Prepare all necessary information so that another group can build a similar board with only the necessary components required for use.

Optional Requirements:

• The user will have the ability to alter the calendar (make appointments) from the unit.
• The unit will be able to implement other calendar systems (Yahoo, Outlook, etc.).
• Adding different user modes in conjunction with Google Calendar. (i.e picture frame mode, other Google apps)
• Interface with a peripheral support (i.e keyboard, USB mouse)

Block Diagram:



Software:

GNU Compiler Collection (GNU is an Computer Operating System composed entirely of free software). It's good because it can be used to compile across different platforms. This was used to build a toolchain and to compile the files that we needed to get linux up an running. The toolchain is to configure the compiler so it has the right information about our hardware.

X-loader is our 1st stage boot loader and is stored in the internal static RAM to initialize memory and enough of the peripheral devices to start the 2nd stage boot loader. This file is name as MLO.bin and stored in the first partition of the Flash Drive for the Beagle board.

U-boot is our 2nd stage boot loader and is used to initialize our OS and peripherals. This can either be stored on the first partition of the Flash Drive, or on the NAND memory of the Beagle board.

A Custom Linux Kernel is the interface between the U-Boot and user interface. This was created with the source code of the 2.6.33.3 kernel. The kernel is configurable to our needs with the menuconfig files (i.e interfacing ethernet, monitor, mouse, how it boots, pretty much anything you can think of in terms of computers). The Kernel is aquired with the GIT server and is updated frequently with the improvements of other community members.

Ubuntu 9.10 is our root file system that we have to work with. This is the top layer of the software that we are exposed to and interact with.

Python is our scripting language that was used to create a user interface, as well as retrieve data from Google.

wxPython-GUI toolkit allowed us to create a graphical environment with python.

Linux Wireless tools. This is a firmware update that allows the Operating System to interface with Ethernet and Wi-Fi.

Project Comments:

Due to the fact that the beagle board is a relatively new offering, in conjunction with a major change in how Ubuntu handles system scripts, development had to be slow and incremental. The lack of documentation on all fronts did not allow us to make any large leaps. Even small tasks took a lot of research and testing and even simple tasks behaved strangely due to our unique hardware.

Linux has less than optimal wireless support, which is complicated by the fact that beagle board is specialized hardware.

Due to the recent major changes to Ubuntu, many of the minor changes we had to make to system scripts became major issues. Documentation on the new system is almost non-existent and is only being implemented in a few varieties of Linux.

Python Bugs, time and date errors, setting the current system date and time as well as making sure that the beagle board had access to the network.

Appendix:

Python Code of the Google Calendar GUI

```
#!/usr/bin/python

import calendar, datetime, wx, struct, time, threading, StringIO, cStringIO, sys,
urllib2

try:
  from xml.etree import ElementTree
except ImportError:
  from elementtree import ElementTree
import gdata.calendar.service
import gdata.service
import atom.service
import gdata.calendar
import atom
import getopt
#import wx.calendar
import wx.lib.sheet

from wx.lib.analogclock import *
from wxPython.lib.sheet import *
#from wx.lib.plot import *


global emailList, Week_List, Time_List, X, Y
global Display_ID, Total_Number_of_Faculty
global Week_Starts, Week_Ends
global Calendar_Feed
global Col_Size, Row_Size, Email_List_Jumper

''' Green, Jake, Mark, '''
Email_List=[
        'Chao You',
            'you.chao@gmail.com',
            'http://www.ndsu.edu/ece/images/people/ChaoYou.gif',
            'ECE 101N',
            '701-231-7402',
            'chao.you@ndsu.edu',
        'Senior Design 0911',
            'sdesign02@gmail.com',
            'http://saturn.ece.ndsu.nodak.edu/ecewiki/images/d/d0/Groupdesign1.jpg',
            'ECE 223',
            '',
```

```
            'SDesign02@gmail.com',
        'Roger Green',
            'f5sogh0sc0ifag5f1lgur0eqs4@group.calendar.google.com',
            'http://www.ndsu.edu/ece/images/people/RogerGreen.gif',
            'ECE 215B',
            '701-231-1024',
            'Roger.Green@ndsu.edu',
        'Jake Glower',
            'ods0jd24hiqhjcscjm13fttsm4@group.calendar.google.com',
            'http://www.ndsu.edu/ece/images/people/JakeGlower.gif',
            'ECE 101B',
            '701-231-7608',
            'Jake.Glower@ndsu.edu',
        'Mark Schroeder',
            'rb1ej450i30thdssphroivrkbs@group.calendar.google.com',
            'http://www.ndsu.edu/ece/images/people/MarkShroeder.gif',
            'ECE 101T',
            '701-231-8049',
            'Mark.Schroeder@ndsu.edu',

        ]
Email_List_Jumper = 6


Week_List = ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday',]
Time_List = ['07:00','07:30','08:00','08:30',
'09:00','09:30','10:00','10:30','11:00','11:30','12:00','12:30','13:00','13:50','14:00','14:30','15:00','15:30','16:00','

Display_ID =0
Total_Number_of_Faculty = 5


ID_COUNT = wx.NewId()
Current_Time = time.localtime()
Week_Starts=Current_Time.tm_mday-Current_Time.tm_wday


if Current_Time.tm_mon == 5 or Current_Time.tm_mon == 7 or Current_Time.tm_mon ==
8
or Current_Time.tm_mon == 12:
    if Week_Starts <= 0:
        Week_Starts = Week_Starts + 30
if Current_Time.tm_mon == 3 and (Current_Time.tm_year)%4 == 0:
    if Week_Starts <= 0:
        Week_Starts = Week_Starts + 29
if Current_Time.tm_mon == 3:
    if Week_Starts <= 0:
        Week_Starts = Week_Starts + 28
```

```
if Current_Time.tm_mon == 2 or Current_Time.tm_mon == 4 or Current_Time.tm_mon ==
6
or Current_Time.tm_mon == 8 or Current_Time.tm_mon == 9 or Current_Time.tm_mon ==
11
or Current_Time.tm_mon == 1:
        if Week_Starts <= 0:
                Week_Starts = Week_Starts + 31

Week_Ends=Week_Starts+6

if Current_Time.tm_mon == 4 or Current_Time.tm_mon == 6 or Current_Time.tm_mon ==
9
or Current_Time.tm_mon == 11:
        if Week_Ends > 30:
                Week_Ends = Week_Ends - 30
if Current_Time.tm_mon == 2 and (Current_Time.tm_year)%4 == 0:
        if Week_Ends > 29:
                Week_Ends = Week_Ends - 29
if Current_Time.tm_mon == 2:
        if Week_Ends > 28:
                Week_Ends = Week_Ends - 28
if Current_Time.tm_mon == 1 or Current_Time.tm_mon == 3 or Current_Time.tm_mon ==
5
or Current_Time.tm_mon == 7 or Current_Time.tm_mon == 8 or Current_Time.tm_mon ==
10
or Current_Time.tm_mon == 12:
        if Week_Ends > 31:
                Week_Ends = Week_Ends - 31

X=len(Week_List)-1
Y=len(Time_List)

myEVT_COUNT = wx.NewEventType()
EVT_COUNT = wx.PyEventBinder(myEVT_COUNT, 1)

class CountEvent(wx.PyCommandEvent):
    """Event to signal that a count value is ready"""
    def __init__(self, etype, eid, value=None):
        """Creates the event object"""
        wx.PyCommandEvent.__init__(self, etype, eid)
        # foo = input("Input something")
        self._value = 1

    def GetValue(self):
        """Returns the value from the event.        """
        return self._value
```

```
class CountingThread(threading.Thread):
    def __init__(self, parent, value):
        """
        @param parent: The gui object that should recieve the value
        @param value: value to 'calculate' to
        """
        threading.Thread.__init__(self)
        self._parent = parent
        self._value = value

    def run(self):
        """Overrides Thread.run. Don't call this directly its called internally
        when you call Thread.start().
        """
        inputDevice = "/dev/event0"    # Set the trigger, which is in the event0.
Code 276 is the USER button

        # format of the event structure (int, int, short, short, int)
        inputEventFormat = 'iihhi'
        inputEventSize = 16
        Event_file = open(inputDevice, "rb") # standard binary file input
        UserButton_event = Event_file.read(inputEventSize)
        (time1, time2, type, code, value) = struct.unpack(inputEventFormat,
UserButton_event)
        if type == 1 and code == 276 and value == 1:
          #print "User button pressed!"
          Display_ID = Display_ID + 1
          if Display_ID >= Total_Number_of_Faculty:
             Display_ID = 0
          #print ' %d ' % Display_ID
          evt = CountEvent(myEVT_COUNT, -1, self._value)
          wx.PostEvent(self._parent, evt)
        self.run()

class MySheet(CSheet):
    instance = 0
    def __init__(self, parent):
        CSheet.__init__(self, parent)
        self.SetLabelBackgroundColour('#DBD4D4')
        self.SetRowLabelAlignment(wx.ALIGN_CENTRE, wx.ALIGN_CENTRE)
        self.text = ''



class Main_Frame(wx.Frame):
    """ class MyFrame inherits wxFrame, uses default ID of -1"""
```

```
    def __init__(self ):

        #Event_file = open(inputDevice, "rb") # standard binary file input
        #UserButton_event = Event_file.read(inputEventSize)
        """  This ID is used to toggle email list. """
        Selection_X = 0             ### Selection list position
        Selection_Width = 200


        ''' Add style=wx.SUNKEN_BORDER  to remove boarder '''
        # wx.Frame.__init__(self, None, -1, 'Calendar', size=(550, 600),
style=wx.SUNKEN_BORDER)
        # wx.Frame.__init__(self, None, -1, 'Calendar',
size=(770,880),style=wx.NO_BORDER)
        #wx.Frame.__init__(self, None, -1, 'Calendar', size=(790,880))
        wx.Frame.__init__(self, None, -1, 'Calendar', size=(1280,1020))
        time.strftime("%A, %B, %Y")
        Current_Time = time.localtime()
        #print (12, 43,23)
        #print Current_Time
        print "Current Time :\t%s" % Current_Time
        Week_Starts=Current_Time.tm_mday-Current_Time.tm_wday
        Week_Ends=Week_Starts+6
        print "this week starts on %d and ends on %d" % (Week_Starts, Week_Ends)
        print time.strftime("%A, %B, %d, %Y %H:%M", Current_Time)


        #Top_box = wx.BoxSizer(wx.VERTICAL)
        System_Box = wx.BoxSizer(wx.VERTICAL)
        self.Info_Panel = wx.Panel(self, -1, style=wx.NO_BORDER, pos=(0,20))
        self.Main_Panel = wx.lib.sheet.CSheet(self)
        self.Clock_Panel = AnalogClockWindow(self, -1, pos=(1000,0), size=(275,275))
        System_Box.Add(self.Info_Panel, 0, wx.EXPAND)
        System_Box.Add(self.Main_Panel, 1, wx.EXPAND)
        self.SetSizer(System_Box)




        ''' ================================='''
        '''         Info  Panel           '''
        ''' ================================='''

        P1_Sizer = wx.BoxSizer(wx.HORIZONTAL)
        self.P1_Photo = wx.Panel(self.Info_Panel,-1, style=wx.NO_BORDER)
        self.P1_List = wx.Panel(self.Info_Panel,-1, style=wx.NO_BORDER)
        self.P1_Label = wx.Panel(self.Info_Panel,-1, style=wx.NO_BORDER)
        self.P1_Clock = wx.Panel(self.Clock_Panel, -1, style=wx.NO_BORDER)
```

```
P1_Sizer.Add(self.P1_Photo,0,wx.EXPAND, 5)
P1_Sizer.Add(self.P1_List ,0,wx.EXPAND, 0)
P1_Sizer.Add(self.P1_Label ,1,wx.EXPAND, 0)
P1_Sizer.Add(self.P1_Clock, 1, wx.ALIGN_RIGHT, 10)
self.Info_Panel.SetSizer(P1_Sizer)
self.Clock_Panel.SetBackgroundColour('YELLOW')
''' Here is the function to draw the picture '''

self.bitmap = wx.StaticBitmap(self.P1_Photo, -1)
Face_url = Email_List[Display_ID+2]
fp = urllib2.urlopen(Face_url)
data = fp.read()
fp.close()
stream = cStringIO.StringIO(data)
Face_bmp = wx.BitmapFromImage( wx.ImageFromStream( stream ))
self.bitmap.SetBitmap(Face_bmp)

#wx.StaticBitmap(self.P1_Photo, -1, Face_bmp, (0, 0))

font = wx.Font(18, wx.ROMAN, wx.NORMAL, wx.NORMAL)

Name_box=wx.StaticText(self.P1_List, -1, "Name: ")
Office_box=wx.StaticText(self.P1_List, -1, "Office: ")
Phone_box=wx.StaticText(self.P1_List, -1, "Phone: ")
Email_box=wx.StaticText(self.P1_List, -1, "Email: ")
Name_box.SetFont(font)
Office_box.SetFont(font)
Phone_box.SetFont(font)
Email_box.SetFont(font)

List_box = wx.BoxSizer(wx.VERTICAL)
List_box.Add(Name_box,0, wx.EXPAND | wx.ALIGN_RIGHT)
List_box.Add(Office_box,0, wx.EXPAND )
List_box.Add(Phone_box,0, wx.EXPAND )
List_box.Add(Email_box,0, wx.EXPAND )
self.P1_List.SetSizer(List_box)

self.Name_Label=wx.StaticText(self.P1_Label, -1, "")
self.Office_Label=wx.StaticText(self.P1_Label, -1, "")
self.Phone_Label=wx.StaticText(self.P1_Label, -1, "")
self.Email_Label=wx.StaticText(self.P1_Label, -1, "")

self.Name_Label.SetFont(font)
self.Office_Label.SetFont(font)
self.Phone_Label.SetFont(font)
self.Email_Label.SetFont(font)

Label_box = wx.BoxSizer(wx.VERTICAL)
```

```
        Label_box.Add(self.Name_Label,0, wx.EXPAND )
        Label_box.Add(self.Office_Label,0, wx.EXPAND )
        Label_box.Add(self.Phone_Label,0, wx.EXPAND )
        Label_box.Add(self.Email_Label,0, wx.EXPAND )
        self.P1_Label.SetSizer(Label_box)


        ''' ================================='''
        '''         Main  Panel           '''
        ''' ================================='''

        self.Main_Panel.SetRowLabelAlignment(wx.ALIGN_CENTRE, wx.ALIGN_CENTRE)
        self.Main_Panel.text = ''
        self.Main_Panel.SetNumberRows(Y)
        self.Main_Panel.SetNumberCols(X+1)
        Col_Size=174
        Row_Size=31
        for ix, title in enumerate(Week_List):
            self.Main_Panel.SetColLabelValue(ix, title)
            self.Main_Panel.SetColSize(ix,Col_Size)
        for iy, title in enumerate(Time_List):
            self.Main_Panel.SetRowLabelValue(iy, title)
            self.Main_Panel.SetRowSize(iy,Row_Size)
        self.P1_List.SetFocus()



        #self.PrintUsage = wx.TextCtrl(self.Top_Panel, -1, 'Select a Calender to
display')
        #self.PrintID = wx.StaticText(self.Top_Panel,  -1, 'Second static box to
display')
        self.Toggle_Button = wx.Button(self.P1_Label, label="Switch Calendars",
pos=(20, 165), size=(220, 45))
        self.Exit_Button = wx.Button(self.P1_Label, label="Exit Calendar",
pos=(270, 165), size=(220, 45))

        self.Toggle_Button.SetFont(wx.Font(16, wx.SWISS, wx.NORMAL, wx.BOLD))
        self.Exit_Button.SetFont(wx.Font(16, wx.SWISS, wx.NORMAL, wx.BOLD))
        self.Toggle_Button.SetBackgroundColour('YELLOW')

        self.Exit_Button.SetBackgroundColour('YELLOW')


        ''' ================================='''
        '''        Other                  '''
        ''' ================================='''
```

```
        ''' ==================================='''
        '''         Event Functions          '''
        ''' ==================================='''
        """     The following is the event happening in the system    """
        self.Toggle_Button.Bind(wx.EVT_BUTTON, self.Button_Event)
        self.Exit_Button.Bind(wx.EVT_BUTTON, self.Exit_Event)
        self.Bind(EVT_COUNT, self.DrawCal)

        """ Enable the following line to open the /dev/event0 trigger """
        #worker = CountingThread(self, 1)
        #worker.start()
        self.Center()
        self.Show(True)
        self.DrawCal()



    def SetStart(self):
        username= Email_List[Display_ID+1]
        visibility = 'public'
        projection = 'full'

        ''' ==================================='''
        '''      Original Calendar Feed     '''
        ''' ==================================='''

        MyCal = gdata.calendar.service.CalendarEventQuery(username, visibility,
projection)
        CurrentTime = time.strftime("%y/%m/%d %H:%M", time.localtime())
        ThisDate = Current_Time.tm_mday
        MyCal.start_min = str(time.strftime("%Y-%m-0", Current_Time))+
str(Week_Starts)
        MyCal.start_max = str(time.strftime("%Y-%m-", Current_Time))+ str(Week_Ends+1)
        MyCal.selectedCalendar='Ok'
        calendar_service = gdata.calendar.service.CalendarService()
        Fetch_Feed = calendar_service.CalendarQuery(MyCal)
        return Fetch_Feed



    def Move_Event(self, event):
        x, y =event.GetPosition()
        x = str(x)
        y = str(y)
        self.SetStatusText("Location (" + x + "," +y + ")")

    def Exit_Event(self, event):
```

```
        self.Close()

    class CalFormat:
        def __init__(self, Time_String):
            self.year=Time_String[:4]
            self.month=Time_String[5:7]
            self.date=Time_String[8:10]
            self.hour=Time_String[11:13]
            self.minute=Time_String[14:16]

    def DrawCal(self):
        TheString = ''
        feed = self.SetStart()
        for i, OneEvent in enumerate(feed.entry):
            TheString = TheString + OneEvent.title.text + '\n'
        for i in range(X+1):
            for j in range(Y):
                self.Main_Panel.SetCellValue(j,i,'')

        Face_url = Email_List[Display_ID+2]
        fp = urllib2.urlopen(Face_url)
        data = fp.read()
        fp.close()
        stream = cStringIO.StringIO(data)
        Face_bmp = wx.BitmapFromImage( wx.ImageFromStream( stream ))
        self.bitmap.SetBitmap(Face_bmp)

        self.Name_Label.SetLabel(Email_List[Display_ID])
        self.Office_Label.SetLabel(Email_List[Display_ID+3])
        self.Phone_Label.SetLabel(Email_List[Display_ID+4])
        self.Email_Label.SetLabel(Email_List[Display_ID+5])




        for i, an_event in zip(xrange(len(feed.entry)), feed.entry):
            print '\t%s.\tTitle:\t\t%s' % (i, an_event.title.text,)
            print '\t\tContent: \t%s' % (an_event.content.text,)

            for a_participant in an_event.who:
                print '\t\tUser Email:\t%s' % (a_participant.email,)
                print '\t\tUser Name: \t%s' % (a_participant.name,)
                if a_participant.attendee_status:
                    print '\t\tStatus (any):\t%s' %
(a_participant.attendee_status.value,)
            for Cal_Time in an_event.when:
                #print '\t\tStart time: \t%s' % (Cal_Time.start_time,)
                #print '\t\tEnd time:   \t%s' % (Cal_Time.end_time,)
```

```
            AA = self.CalFormat(Cal_Time.start_time)
            print '\t\tStart time: \t%s-%s-%s at time %s:%s %s' % (AA.year,
AA.month, AA.date, AA.hour, AA.minute,
Week_List[int(AA.date)-Week_Starts+1])
            AA = self.CalFormat(Cal_Time.end_time)
            print '\t\tEnd time: \t%s-%s-%s at time %s:%s' % (AA.year, AA.month,
AA.date, AA.hour, AA.minute)
            Select_Col = int(AA.date)-Week_Starts
            if AA.hour=="":
               Hour_String="08:00"
               Select_Row =Time_List.index(Hour_String)-2
            else:
               Hour_String=AA.hour+":00"
               Select_Row =Time_List.index(Hour_String)-2
            print '\t\tThe Index will be:\t %d %d' % (Select_Row, Select_Col)

            self.Main_Panel.SetCellValue(Select_Row, Select_Col,
an_event.title.text)

         for Cal_Loca in an_event.where:
            print '\t\tLocation: \t%s' % (Cal_Loca.value_string,)
         print '\n\n'

   def Selection_Event(self, event):
      "'get the selected colour choice'"
      Display_ID = Display_ID + Email_List_Jumper
      if Display_ID >=Total_Number_of_Faculty*Email_List_Jumper:
         Display_ID = 0
      #print ' %d ' % Display_ID
      self.DrawCal()
   #self.Refresh()

   def Button_Event(self, event):
      global Display_ID, Total_Number_of_Faculty
      Display_ID = Display_ID + Email_List_Jumper
      if Display_ID >= Total_Number_of_Faculty*Email_List_Jumper:
         Display_ID = 0
      #print ' %d ' % Display_ID
      self.DrawCal()



app = wx.App(False)  # Create a new app, don't redirect stdout/stderr to a window.
#print "Hello World"
frame = Main_Frame() # A Frame is a top-level window.
""" Bring the window out
"""
app.MainLoop()
```

```
sys.exit(True)
```